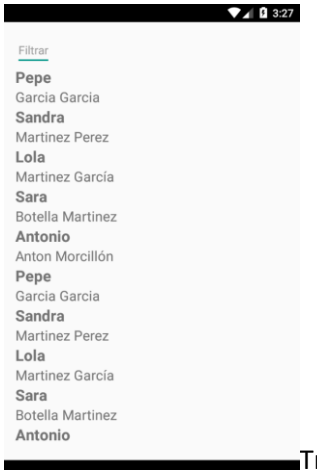
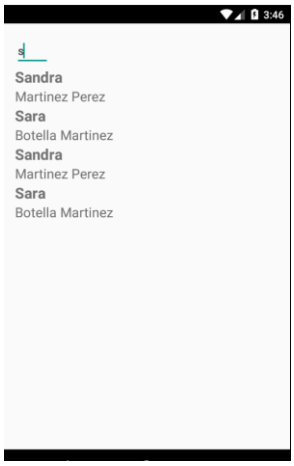
Vamos a realizar una aplicación que nos permita buscar o filtrar los elementos de un recycler. El aspecto visual de la misma es el siguiente:



Tras pulsar en el *EditText* con el carácter S



Lo primero que tenemos que tener en cuenta es que vamos a manejar una lista donde cada elemento está formado por el apellido y el nombre de un contacto. Para ello definiremos una clase que llamaremos *DatosAlumno.java*:

```
class DatosAlumno {
    private  String Nombre;
    private String Apellidos;
    private int Edad;
    public  DatosAlumno(String nombre,String apellidos, int edad )
    {
        Nombre=nombre;
        Apellidos=apellidos;
        Edad=edad;
    }
    public DatosAlumno(String nombre,String apellidos ) {
        Nombre = nombre;
        Apellidos = apellidos;
    }

    public String getNombre() { return Nombre; }

    public String getApellidos() { return Apellidos; }
}
```

Como ya sabemos tenemos que definir también un layout que definirá los elementos a mostrar en la línea del recycler:

```
<LinearLayout xmlns:android="http://schemas.android.com/a
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="wrap_content">
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Nombre"
        android:textSize="20dp"
        android:textStyle="bold"
        android:id="@+id/textViewNombre" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Apellidos"
        android:textSize="18dp"
        android:id="@+id/textViewApellidos" />

</LinearLayout>
```

El layout principal de la aplicación vendrá a ser similar a éste:

```
<RelativeLayout xmlns:android="http://schemas.android.co
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="16dp"
    android:paddingRight="16dp"
    android:paddingTop="16dp"
    android:paddingBottom="16dp">
    <EditText
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/editText"
        android:textSize="15dp"
        android:hint="Filtrar" />

    <android.support.v7.widget.RecyclerView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/recycler"
        android:layout_below="@+id/editText" />

</RelativeLayout>
```

Como vemos contendrá un EditText y un Recycler. Es importante que analicemos la propiedad android:hint del editText, ya que permitirá que el control tenga un fondo, que desaparecerá cuando pulsemos sobre el control.

Tendremos definido una clase Adaptador y una clase Holder, tal y como hemos visto durante este tema:

```
public class Adaptador extends RecyclerView.Adapter implements View.OnClickListener{
    Context context;
    View.OnClickListener listener;
    public Adaptador(Context context) {
        this.context=context;  }
    @Override
    public RecyclerView.ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
        View vistaAlumno =LayoutInflater.from(parent.getContext()).inflate(R.layout.datos_alumnios, parent,false);
        vistaAlumno.setOnClickListener(this);
        return new Holder(vistaAlumno);
    }
    @Override
    public void onBindViewHolder(RecyclerView.ViewHolder holder, int position) {
        ((Holder)holder).bind(((MyActivity)context).datosAlumno.get(position));
    }
    @Override
    public int getItemCount() {
        return ((MyActivity)context).datosAlumno.size();
    }
    public void clickListener(View.OnClickListener listener)
    {
        this.listener=listener;
    }
    @Override
    public void onClick(View view) {
        if(listener!=null) listener.onClick(view);
    }
}
```

```
public class Holder extends RecyclerView.ViewHolder {
    TextView nombre, apellido;
    public Holder(View itemView) {
        super(itemView);
        nombre=itemView.findViewById(R.id.textViewNombre);
        apellido=itemView.findViewById(R.id.textViewApellidos);
    }
    void bind(DatosAlumno datosAlumno) {
        nombre.setText(datosAlumno.getNombre());
        apellido.setText(datosAlumno.getApellidos());
    }
}
```

También definiremos los pasos necesarios para implementar el recycler con el onClick en la actividad principal:

```
public class MyActivity extends Activity implements View.OnClickListener {
    ArrayList<DatosAlumno> datosAlumno;
    RecyclerView lista;
    TextView texto;
    Adaptador adaptador;
    ArrayList<DatosAlumno>  subListaAux;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_my);
        cargarDatos();
        subListaAux = datosAlumno;
        lista = findViewById(R.id.recycler);
        texto = findViewById(R.id.editText);
        adaptador = new Adaptador(this);;
        adaptador.clickListener(this);
        lista.setAdapter(adaptador);
        lista.setLayoutManager(new LinearLayoutManager(this,LinearLayoutManager.VERTICAL,false));
        texto.addTextChangedListener(new TextWatcher() {...});
    }
    void cargarDatos()
    {...}
    @Override
    public void onClick(View view) {
        Toast t = Toast.makeText(this, "Has Seleccionado a " +
          datosAlumno.get(lista.getChildAdapterPosition(view)).getNombre(), Toast.LENGTH_LONG);
        t.show();
    }
}
```

Finalmente tenemos que implementar el cómo realizar el filtrado:

```
texto.addTextChangedListener(new TextWatcher() {
    @Override
    public void beforeTextChanged(CharSequence s, int start, int count, int after) {...}

    @Override
    public void onTextChanged(CharSequence s, int start, int before, int count) {

        ArrayList<DatosAlumno> subLista = new ArrayList<DatosAlumno>();
        datosAlumno = subListaAux;
        lista.setAdapter(adaptador);
        if (!TextUtils.isEmpty((texto.getText().toString()))) {
            int longitud = texto.getText().toString().length();
            for (DatosAlumno x : datosAlumno)
                try {
                    if (texto.getText().toString().compareToIgnoreCase(x.getNombre().substring(0, longitud)) == 0)
                        subLista.add(x);
                }catch (StringIndexOutOfBoundsException e){}
            datosAlumno = subLista;
            lista.setAdapter(adaptador);
        }
    }
    @Override
    public void afterTextChanged(Editable s) {...}
});
```

Lo que realmente hacemos es mantener dos listas diferentes, una con la lista completa y otra que se irá actualizando según se realice el filtrado. El código es muy sencillo tan solo configuraremos un *listener* que responderá a los cambios de texto en el *EditTex*t recargando el *adapter* de la lista con los nuevos datos, para ello tenemos que programar el método *onTextChanged()*. Recorremos la lista y para cada dato cuya longitud es mayor que la longitud de la cadena del *EditText* comprobamos si el string coincide con el de la lista (hasta la longitud que tiene el filtro).

Realizar el filtrado tanto para el nombre como el apellido en los dos supuestos.